From Quantified CTL to QBF

Akash Hossain
IRIF, Univ. Paris Diderot, France
akashoss42@gmail.com

François Laroussinie
IRIF, Univ. Paris Diderot, France
francois@irif.fr

Abstract

QCTL extends the temporal logic CTL with quantifications over atomic propositions. This extension is known to be very expressive: QCTL allows us to express complex properties over Kripke structures (it is as expressive as MSO). Several semantics exist for the quantifications: here, we work with the structure semantics, where the extra propositions label the Kripke structure (and not its execution tree), and the model-checking problem is known to be PSPACE-complete in this framework. We propose a model-checking algorithm for QCTL based on a reduction to QBF. We consider several reduction strategies, and we compare them with a prototype (based on the SMT-solver Z3) on several examples.

1 Introduction

Temporal logics have been introduced in computer science in the late 1970’s [14]; they provide a powerful formalism for specifying correctness properties of evolving systems. Various kinds of temporal logics have been defined, with different expressiveness and algorithmic properties. For instance, the Computation Tree Logic (CTL) expresses properties of the computation tree of the system under study (time is branching: a state may have several successors), and the Linear-time Temporal Logic (LTL) expresses properties of one execution at a time (a system is viewed as a set of executions).

Temporal logics allow model checking, i.e. the automatic verification that a finite state system satisfies its expected behavioral specifications [15, 3]. It is well known that CTL model-checking is PTIME-complete and LTL model-checking (based on automata techniques) is PSPACE-complete. Verification tools exist for both logics and model-checking is now commonly used in the design of critical reactive systems. The main limitation to this approach is the state-explosion problem: symbolic techniques (for example with BDD), SAT-based approaches, or partial order reductions have been developed and they are impressively successful. The SAT-based model-checking consists in using SAT-solvers in the decision procedures. It was first developed for bounded model-checking (to search for executions whose length is bounded by some integer, satisfying some temporal property) which can be reduced to some satisfiability problem and then can be solved by a SAT-solver [2]. SAT approaches have also been extended to unbounded verification and combined with other techniques [12]. Many studies have been done in this area, and it is widely considered as
an important approach in practice, which complements other symbolic techniques like BDD (see [1] for a survey).

In terms of expressiveness, CTL (or LTL) still has some limitations: in particular, it lacks the ability of counting. For instance, it cannot express that an event occurs (at least) at every even position along a path, or that a state has two successors. In order to cope with this, temporal logics have been extended with propositional quantifiers [16]: those quantifiers allow for adding fresh atomic propositions in the model before evaluating the truth value of a temporal-logic formula. That a state has at least two successors can then be expressed (in quantified CTL, hereafter written QCTL) by saying that it is possible to label the model with atomic proposition $p$ in such a way that there is a successor that is labelled with $p$ and one that is not.

Different semantics for QCTL have been studied in the literature depending on the definition of the labelling: either it refers to the finite-state model – it is the structure semantics – or it refers to the execution tree – it is the tree semantics. Both semantics are interesting and have been extensively studied [9, 7, 13, 8, 4, 10]. While the tree semantics allow us to use the tree automata techniques to get decision procedures (model-checking and satisfiability are TOWER-complete [10]), the situation is quite different for the structure semantics: in this framework, model-checking is PSPACE-complete and satisfiability is undecidable [7].

In this paper, we focus on the structure semantics, and we propose a model-checking algorithm based on a reduction to QBF (propositional logic augmented with quantifiers): given a Kripke structure $K$ and a QCTL formula $\Phi$, we show how to build a QBF formula $\hat{\Phi}_K$ which is valid iff $K \models \Phi$. It is natural to use QBF quantifiers to deal with propositional quantifiers of QCTL. Of course, QBF-solvers are not as efficient as SAT-solvers, but still much progress has been made (and QBF-solvers have already been considered for model-checking, as in [6]). We propose several reductions depending on the way of dealing with nested temporal modalities, and we have implemented a prototype (based on Z3 SMT-solver [5]) to compare these reductions over several examples. As far as we know, it is the first implementation of a model-checker for QCTL.

Here, our first objective is to use the QBF-solver as a tool to check complex properties over limited size models, and this is therefore different from the classical use of SAT-based techniques which are precisely applied to solve verification problems for very large systems.

The outline of the paper is as follows: we begin with setting up the necessary formalism in order to define QCTL. We then devote Section 3 to the different reductions to QBF. Finally, Section 4 contains several practical results and examples.

## 2 Definitions

### 2.1 Kripke structures

Let $AP$ be a set of atomic propositions.

> **Definition 1.** A Kripke structure is a tuple $K = (V, E, \ell)$, where $V$ is a finite set of vertices (or states), $E \subseteq V \times V$ is a set of edges (we assume that for any $x \in V$, there exists $x' \in V$ s.t. $(x, x') \in E$), and $\ell : V \to 2^{AP}$ is a labelling function.

An infinite path (also called an execution) in a Kripke structure is an infinite sequence $\rho = x_0x_1x_2 \ldots$ such that for any $i$ we have $x_i \in V$ and $(x_i, x_{i+1}) \in E$. We write $\text{Path}_K^\omega$ for the set of infinite paths of $K$ and $\text{Path}_K^\omega(x)$ for the set of infinite paths issued from $x \in V$. 

Given such a path $\rho$, we use $\rho_{\leq i}$ to denote the $i$-th prefix $x_0 \ldots x_i$, $\rho_{\geq i}$ for the $i$-th suffix $x_i x_{i+1} \ldots$, and $\rho(i)$ for the vertex $x_i$. The size of $K$ is $|V| + |E|$.

Given a set $P \subseteq AP$, two Kripke structures $K = (V, E, \ell)$ and $K' = (V', E', \ell')$ are said $P$-equivalent (denoted by $K \equiv_P K'$) if $V = V'$, $E = E'$, and for every $x \in V$ we have: $\ell(x) \cap P = \ell'(x) \cap P$.

### 2.2 QCTL

This section is devoted to the definition of the logic QCTL, which extends the classical branching-time temporal logic CTL with quantifications over atomic propositions.

| Definition 2. | The syntax of QCTL is defined by the following grammar: |
| QCTL $\exists \varphi, \psi ::= q \mid \neg \varphi \mid \varphi \lor \psi \mid \text{EX} \varphi \mid \text{EU} \psi \mid \text{A} \varphi \text{U} \psi \mid \exists p. \varphi$ |

where $q$ and $p$ range over $AP$.

QCTL formulas are evaluated over states of Kripke structures:

| Definition 3. | Let $K = (V, E, \ell)$ be a Kripke structure, and $x \in V$. The semantics of QCTL formulas is defined inductively as follows: |

$K, x \models p$ iff $p \in \ell(x)$

$K, x \models \neg \varphi$ iff $K, x \not\models \varphi$

$K, x \models \varphi \lor \psi$ iff $K, x \models \varphi$ or $K, x \models \psi$.

$K, x \models \text{EX} \varphi$ iff $\exists (x, x') \in E$ s.t. $K, x' \models \varphi$

$K, x \models \text{EU} \psi$ iff $\exists p \in \text{Path}^K_\leq(x), \exists i \geq 0$ s.t. $K, \rho(i) \models \psi$ and for any $0 \leq j < i$, we have $K, \rho(j) \models \varphi$

$K, x \models \text{A} \varphi \text{U} \psi$ iff $\forall p \in \text{Path}^K_\leq(x), \exists i \geq 0$ s.t. $K, \rho(i) \models \psi$ and for any $0 \leq j < i$, we have $K, \rho(j) \models \varphi$

$K, x \models \exists p. \varphi$ iff $\exists K' \equiv_{AP \setminus \{p\)} K$ s.t. $K', x \models \varphi$

In the sequel, we use standard abbreviations such as $\top, \bot, \land, \lor$ and $\Leftrightarrow$. We also use the additional (classical) temporal modalities of CTL: $A \varphi = \neg \text{EX} \neg \varphi$, $E \varphi = \text{ET} \text{U} \varphi$, $A \varphi = A \top \text{U} \varphi$, $E \varphi = \neg A \text{F} \neg \varphi$, $A \varphi = \neg E \text{F} \neg \varphi$, $E \varphi \text{W} \psi = \neg A \neg \psi \text{U} (\neg \psi \land \neg \varphi)$ and $A \varphi \text{W} \psi = \neg E \neg \psi \text{U} (\neg \psi \land \neg \varphi)$.

Moreover, we use the following abbreviations related to quantifiers over atomic propositions: $\forall p. \varphi = \neg \exists p. \neg \varphi$, and for a set $P = \{p_1, \ldots, p_k\} \subseteq AP$, we write $\exists P. \varphi$ for $\exists p_1. \ldots \exists p_k. \varphi$ and $\forall P. \varphi$ for $\forall p_1. \ldots \forall p_k. \varphi$.

The size of a formula $\varphi \in$ QCTL, denoted $|\varphi|$, is defined inductively by: $|q| = 1$, $|\neg \varphi| = |\exists p. \varphi| = |\text{EX} \varphi| = 1 + |\varphi|$, $|\varphi \lor \psi| = |\text{EU} \psi| = |A \varphi \text{U} \psi| = 1 + |\varphi| + |\psi|$. Moreover we use $\text{ht}(\varphi)$ to denote the temporal height of $\varphi$, that is the maximal number of nested temporal modalities in $\varphi$. And given a subformula $\psi$ in $\Phi$, the temporal depth of $\psi$ in $\Phi$ (denoted $\text{td}_\varphi(\psi)$) is the number of temporal modalities having $\psi$ in their scope.

In the following, we denote by SubF($\Phi$) (resp. SubTF($\Phi$)) the set of subformulas of $\Phi$ (resp. the set of subformulas starting with a temporal modality).

Two QCTL formulas $\varphi$ and $\psi$ are said to be equivalent (written $\varphi \equiv \psi$) iff for any structure $K$, any state $x$, we have $K, x \models \varphi$ iff $K, x \models \psi$. This equivalence is substitutive.
2.2.0.1 Discussion on the semantics.

The semantics we defined is classically called the structure semantics: a formula $\exists p. \varphi$ holds true in a Kripke structure $K$ if there exists a $p$-labelling of the structure $K$ such that $\varphi$ is satisfied. Another well-known semantics coexists in the literature for propositional quantifiers, the tree semantics: $\exists p. \varphi$ holds true when there exists a $p$-labelling of the execution tree (the infinite unfolding) of the Kripke structure under which $\varphi$ holds. If, for CTL, interpreting formulas over the structure or the execution tree is equivalent, this is not the case for QCTL. Moreover, these two semantics do not have the same algorithmic properties: if QCTL model-checking and satisfiability are TOWER-complete for the tree semantics (the algorithms are based on tree automata techniques), QCTL model-checking is PSPACE-complete for the structure semantics but satisfiability is undecidable. (see [10] for a survey). Nevertheless, in both semantics, QCTL and QCTL* (the extension of CTL* with quantifications) are equally expressive, and are as expressive $^1$ as the Monadic Second-Order Logic over the finite structure or the infinite trees (depending on the semantics). Note also that any QCTL formula is equivalent to a formula in Prenex normal form (we will use this result in next sections).

Finally, there is also the amorphous semantics [7], where $\exists p. \varphi$ holds true at a state $s$ in some Kripke structure $K$ if, and only if, there exists some Kripke structure $K'$ with a state $s'$ such that $s$ and $s'$ are bisimilar, and for which there exists a $p$-labelling making $\varphi$ hold true at $s'$. With these semantics, the logic is insensitive to unwinding, and more generally it is bisimulation-invariant (contrary to the two previous semantics, see below).

2.3 Examples of QCTL formulas

QCTL allows us to express complex properties over Kripke structures: for example, we can build a characteristic formula (up to isomorphism) of a structure, we can reduce model-checking problems for multi-player games to QCTL model-checking [11]... Below, we give several examples of counting properties, to illustrate the expressive power of propositional quantifiers.

The first one of the formulas below expresses that there exists a unique reachable state satisfying $\varphi$, and the second one states that there exists a unique immediate successor satisfying $\varphi$:

\begin{align}
E=1 F \varphi &= EF \varphi \land \forall p. (EF(p \land \varphi) \Rightarrow AG(\varphi \Rightarrow p)) \\
E=1 X \varphi &= EX \varphi \land \forall p. (AX(\varphi \Rightarrow p) \lor AX(\varphi \Rightarrow \neg p))
\end{align}

where we assume that $p$ does not appear in $\varphi$. Consider the formula 1: if there were two reachable states satisfying $\varphi$, then labelling only one of them with $p$ would falsify the $AG$ subformula. For 2, the argument is similar.

The existence of at least $k$ successors satisfying a given property can be expressed with:

$$
E \geq k X \varphi = \exists P. \left( \bigwedge_{1 \leq i \leq k} EX(p_i \land \neg p_i) \land AX \left( \bigvee_{1 \leq i \leq k} p_i \Rightarrow \varphi \right) \right)
$$

And we can define $E=k X \varphi$ as $E \geq k X \varphi \land \neg E \geq k+1 X \varphi$. Note that these examples show why QCTL formulas are not bisimulation-invariant.

$^1$ This requires adequate definitions, since a temporal logic formula may only deal with the reachable part of the model, while MSO has a more global point of view.
When using QCTL to specify properties, one often needs to quantify (existentially or universally) over one reachable state we want to mark with a given atomic proposition. To this aim, we add the following abbreviations:

\[ \exists^1 p. \varphi = \exists p. \left((\mathsf{E}_1 \mathsf{F} p) \land \varphi\right) \quad \forall^1 p. \varphi = \forall p. \left((\mathsf{E}_1 \mathsf{F} p) \Rightarrow \varphi\right) \]

## 3 Model-checking QCTL

Model-checking QCTL is a PSPACE-complete problem (for detailed results about program complexity and formula complexity, see [10]), and it is NP-complete for the restricted set of formulas of the form \( \exists P. \varphi \), with \( P \subseteq \mathsf{AP} \) and \( \varphi \in \mathsf{CTL} \) [9]. In this section, we give a reduction from the QCTL model-checking problem to QBF.

In the following, we assume a Kripke structure \( \mathcal{K} = (V, E, \ell) \), an initial state \( x_0 \in V \) and a QCTL formula \( \Phi \) to be fixed. Let \( V \) be \( \{x_0, \ldots, x_n\} \). We also assume w.l.o.g. that every quantifier \( \exists \) and \( \forall \) in \( \Phi \) introduces a fresh atomic proposition, and distinct from the propositions used in \( \mathcal{K} \). We use \( \mathsf{AP}^\Phi_Q \) to denote the set of quantified atomic propositions in \( \Phi \).

These assumptions allow us to use an alternative notation for the semantics of \( \Phi \)-subformulas: the truth value of \( \varphi \) will be defined for a state \( x \) in \( \mathcal{K} \) within an environment \( \varepsilon : \mathsf{AP}^\Phi_Q \rightarrow 2^V \), that is a partial mapping associating a subset of vertices to a proposition in \( \mathsf{AP}^\Phi_Q \). We use \( \mathcal{K}, x \models \varepsilon \varphi \) to denote that \( \varphi \) holds at \( x \) in \( \mathcal{K} \) within \( \varepsilon \). This ensures that the \( \mathcal{K} \)'s labelling \( \ell \) is not modified when a subformula is evaluated, only \( \varepsilon \) is extended with labellings for new quantified propositions. Formally the main changes of the semantics are as follows:

\[
\mathcal{K}, x \models \varepsilon p \iff \left((p \in \mathsf{AP}^\Phi_Q \text{ and } x \in \varepsilon(p)) \text{ or } (p \notin \mathsf{AP}^\Phi_Q \text{ and } p \in \ell(x))\right) \\
\mathcal{K}, x \models \varepsilon \exists p. \varphi \iff \exists V' \subseteq V \text{ s.t. } \mathcal{K}, x \models \varepsilon[p \mapsto V'] \varphi
\]

where \( \varepsilon[p \mapsto V'] \) denotes the mapping which coincides with \( \varepsilon \) for every proposition in \( \mathsf{AP}^\Phi_Q \setminus \{p\} \) and associates \( V' \) to \( p \).

We use this new notation in order to better distinguish initial \( \mathcal{K} \)'s propositions and quantified propositions to make proofs simpler. Of course, there is no semantic difference: \( \mathcal{K}, x \models \Phi \iff \mathcal{K}, x \models \emptyset \Phi \).

In next sections, we consider general quantified propositional formulas (QBF) of the form:

\[
\mathsf{QBF} \ni \alpha, \beta ::= q \mid \alpha \land \beta \mid \alpha \lor \beta \mid \neg \alpha \mid \alpha \leftrightarrow \beta \mid \exists q. \alpha \mid \forall q. \alpha
\]

The formal semantics of a formula \( \alpha \) is defined over a Boolean valuation for free variables in \( \alpha \) (i.e. propositions which are not bound by a quantifier \(^2\)), and it is defined as usual. A formula is said to be closed when it does not contain free variables. In the following, we use the standard notion of validity for closed QBF formulas.

Our aim is then to build a (closed) QBF formula \( \hat{\Phi}^{x_0} \) such that \( \hat{\Phi}^{x_0} \) is valid iff \( \Phi \) holds true at \( x_0 \) in \( \mathcal{K} \).

### 3.1 Overview

We present several reductions of QCTL model-checking problem to QBF validity problem. Given a \( \Phi \)-subformula \( \varphi \), a vertex \( x \in V \), and a subset \( P \subseteq \mathsf{AP}^\Phi_Q \), we define a QBF formula

\(^2\) We assume w.l.o.g. that every quantifier \( \exists \) and \( \forall \) introduces a new proposition.
\( \neg \phi_{x,P} = \neg \phi_{x,P} \)

\( \phi \lor \psi_{x,P} = \phi_{x,P} \lor \psi_{x,P} \)

\( \text{EX} \phi_{x,P} = \bigvee_{(x,x') \in E} \phi_{x',P} \)

\( \exists p \phi = \exists p^{x_0} \ldots p^{x_n} \phi \cup \{p\} \)

\( p^x = \begin{cases} p^x & \text{if } p \in P \\ \top & \text{if } p \notin P \text{ and } p \in \ell(x) \\ \bot & \text{otherwise} \end{cases} \)

Table 1 Reduction for basic modalities for reductions UU and FP

\[ \begin{align*}
\widehat{E} \varphi U \psi_{x,P} &= \widehat{E} \varphi U \psi_{x,P,\{x\}} & \text{with:} \\
\text{EE} \varphi U \psi^{x,P,X} &= \widehat{\psi}_{x,P} \lor \left( \widehat{\phi}_{x,P} \land \bigvee_{(x,x') \in E \text{ s.t. } x' \notin X} \text{EE} \varphi U \psi^{x',P,X \cup \{x'\}} \right) & \text{(4)} \\
\widehat{A} \varphi U \psi^{x,P,X} &= \widehat{A} \varphi U \psi^{x,P,\{x\}} & \text{with:} \\
\text{AE} \varphi U \psi^{x,P,X} &= \begin{cases} \widehat{\psi}_{x,P} & \text{if } \exists (x, x') \in E \text{ s.t. } x' \in X \\ \widehat{\phi}_{x,P} \lor \left( \widehat{\psi}_{x,P} \land \bigwedge_{(x,x') \in E} \text{AE} \varphi U \psi^{x',P,X \cup \{x'\}} \right) & \text{otherwise} \end{cases} & \text{(5)} \\
\end{align*} \]

Table 2 Reduction for temporal modalities EU and AU – variant UU

\( \widehat{\phi}^{x,P} \) whose variables belong to \( \text{AP}_Q \times V \) (in the following, we use the notation \( p^x \) for \( p \in \text{AP}_Q \) and \( x \in V \)). The first two reductions are based on different encodings of temporal modalities, but share a common part given in Table 1.

### 3.1.1 Unfolding characterization of the until operators

First, we can complete previous construction rules of Table 1 with those of Table 2 to get the first method (called UU). This is a naive approach consisting in encoding the temporal modalities as unfoldings of the transition relation. The rules of Table 2 can be seen as a depth-first way to look for a path satisfying an Until modality (for EU) or its negation (for AU) among all the simple paths issued from \( x \).

Before stating the correctness of the construction, we need to associate a Boolean valuation \( v_x \) for variables in \( \text{AP}_Q \times V \) to an environment \( \varepsilon \) for \( \text{AP}_Q \). We define \( v_x \) as follows: for any \( p \in \text{AP}_Q \) and \( x \in V \), \( v_x(p^x) = \top \) iff \( x \in \varepsilon(p) \).

Now we have the following theorem whose proof is in appendix A:

**Theorem 4.** Given a QCTL formula \( \Phi \), a Kripke structure \( K = \langle V, E, \ell \rangle \), a state \( x \in V \), an environment \( \varepsilon : \text{AP}_Q \rightarrow 2^V \) and a \( \Phi \)-subformula \( \varphi \), if \( \widehat{\phi}_{x,\text{dom}(\varepsilon)}^{x,\varepsilon} \) is defined inductively w.r.t. the rules of Tables 1 and 2, we have: \( K, x \models_{\varepsilon} \varphi \iff v_x \models \widehat{\phi}_{x,\text{dom}(\varepsilon)}^{x,\varepsilon} \)

It remains to define \( \widehat{\Phi}^x \) as \( \widehat{\phi}^x \Phi \) and we get the reduction: \( K, x_0 \models \Phi \iff \widehat{\Phi}^{x_0} \) is valid.

The main drawback of this reduction is the size of the QBF formula: indeed any Until modality may induce a formula whose size is in \( O(|V|!) \), and the size of the resulting formula
A. Hossain and F. Laroussinie

$\Phi^\pi$ is then in $O(|\Phi| \cdot |V|!^{ht(\Phi)})$. Nevertheless, one can notice that the reduction does not use new quantified propositions to encode the temporal modalities, contrary to other methods we will see later.

### 3.2 Fixed point characterization of the until operators

Here we present the fixed point method (called $\text{FP}$) for dealing with the modalities $\mathbf{AU}$ and $\mathbf{EU}$. Let $\varphi$ and $\psi$ be two $\text{QCTL}$-formulas. The idea of the method is to build a $\text{QCTL}$ formula that is equivalent to $E\varphi U \psi$ (or $A\varphi U \psi$), using only the modalities $\mathbf{EX}$, $\mathbf{AX}$ and $\mathbf{AG}$.

We first have the following lemma:

**Lemma 5.** For any $\text{QCTL}$ formula $E\varphi U \psi$, we have:

$$E\varphi U \psi \equiv \forall z. (\mathbf{AG}(z \Leftrightarrow (\psi \lor (\varphi \land \mathbf{EX} z))) \Rightarrow z)$$

**Proof.** Let $x$ be a state in a Kripke structure $\mathcal{K}$. Let $\theta$ be the formula $(\mathbf{AG}(z \Leftrightarrow (\psi \lor (\varphi \land \mathbf{EX} z))) \Rightarrow z)$. Assume $\mathcal{K}, x \models E\varphi U \psi$. We can use the standard characterization of $\mathbf{EU}$ as fixed point: $x$ belongs to the least fixed point of the equation $Z = \psi \lor (\varphi \land \mathbf{EX} Z)$ where $\psi$ (resp. $\varphi$) is here interpreted as the set of states satisfying $\psi$ (resp. $\varphi$). Therefore any $z$-labelling of reachable states from $x$ corresponding to a fixed point will have the state $x$ labelled. This is precisely what is specified by the $\text{QCTL}$ formula.

Now if $\mathcal{K}, x \models \theta$ for every $z$-labelling corresponding to a fixed point of the previous equation, this is the case for the $z$-labelling of the states reachable from $x$ and satisfying $E\varphi U \psi$, and we deduce $\mathcal{K}, x \models E\varphi U \psi$. ◼

And we have the same result for $\mathbf{AU}$:

**Lemma 6.** For any $\text{QCTL}$ formula $A\varphi U \psi$, we have:

$$A\varphi U \psi \equiv \forall z. (\mathbf{AG}(z \Leftrightarrow (\psi \lor (\varphi \land \mathbf{AX} z))) \Rightarrow z)$$

**Proof.** Similar to the proof of Lemma 5 ◼

As a direct consequence, we get the following result:

**Proposition 7.** For any $\text{QCTL}$ formula $\Phi$, we can build an equivalent $\text{QCTL}$ formula $\text{fpc}(\Phi)$ such that: (1) $\text{fpc}(\Phi)$ is built up from atomic propositions, Boolean operators, propositional quantifiers and modalities $\mathbf{EX}$ and $\mathbf{AG}$, and (2) the size of $\text{fpc}(\Phi)$ is linear in $|\Phi|$.

Note that the size of $\text{fpc}(\Phi)$ comes from the fact that there is no duplication of subformulas when applying the transformation rules based on equivalences of Lemmas 5 and 6. Moreover, the temporal height of $\text{fpc}(\Phi)$ is smaller than $ht(\Phi) + 1^4$.

And to translate $\text{fpc}(\Phi)$ into $\text{QBF}$, it remains to add a single rule $^5$ to the definitions of Table 1 to deal with $\mathbf{AG}$:

$$\overline{\mathbf{AG}}\varphi^x = \bigwedge_{(x,y) \in E^*} \overline{\varphi}^y$$  \hspace{1cm} (8)

Where $E^*$ is the reflexive and transitive closure of $E$. Then we have:

---

$^3$ Labelling other states does not matter.

$^4$ The temporal height will be increased by 1 if $\Phi$ has an until operator whose members are Boolean combinations of atomic propositions.

$^5$ For $\mathbf{AX}$ we can either change the rule for $\mathbf{EX}$ with a disjunction, or express it with $\mathbf{EX}$ and $\neg$. 

---
Theorem 8. Given a QCTL formula $\Phi$, a Kripke structure $K = (V,E,\ell)$, a state $x \in V$ and an environment $\varepsilon : AP^x_\Phi \rightarrow 2^V$, for any $\Phi$-subformula $\varphi$, if $\text{fpc}(\varphi)$ is defined inductively w.r.t. the rules of Table 1 and the rule 8, we have:

$$K, x \models \varphi \iff \varepsilon \models \text{fpc}(\varphi)$$

Proof. The proof is a direct consequence of Proposition 7.

With this approach, the size of $\text{fpc}(\Phi)$ is in $O(|\Phi| \cdot |K|^{\text{ht}(\Phi)})$. Indeed, an EU (or AU) modality gives rise to a QBF formula of size $(|V| + |E|)$. The exponential size comes from the potential nesting of temporal modalities: to avoid it, one could consider the DAG-size of formulas. In the next section, we consider another solution. Note also that the number of propositional variables in the QBF formula is bounded by $|AP^x_\Phi| \cdot |V|$.

3.3 Reduction via flat formulas

To avoid the size explosion of $\hat{\Phi}^x$, one can use an alternative approach for Prenex QCTL formulas. Remember that any QCTL formula can be translated into an equivalent QCTL formula in Prenex normal form whose size is linear in the size of the original formula [10].

In the following, a CTL formula is said to be basic when it is of the form $\text{EX}\alpha$, $\text{EA}\alpha \text{U}\beta$ or $\text{A}\alpha \text{U}\beta$ where $\alpha$ and $\beta$ are Boolean combinations of atomic propositions. It is easy to observe that any CTL formula can be translated into a QCTL formula with a temporal height less or equal to 2:

Proposition 9. Any CTL formula $\Phi$ is equivalent to some QCTL formula $\Psi$ of the form:

$$\exists \{\kappa_1 \ldots \kappa_m\}. (\Phi_0 \land \bigwedge_{i=1 \ldots m} \text{AG}(\kappa_i \leftrightarrow \theta_i))$$

where $\Phi_0$ is a Boolean combination of basic CTL formulas and every $\theta_i$ is a basic CTL formula (for any $1 \leq i \leq m$). Moreover, $|\Psi|$ is in $O(|\Phi|)$.

Proof. Let $S_\Phi$ be the set of temporal subformulas occurring in $\Phi$ at a temporal depth greater or equal to 1. We will prove the proposition by induction over the size of $S_\Phi$. If $|S_\Phi| = 0$, the formula satisfies the property. Now consider a formula with $|S_\Phi| > 0$. $\Phi$ must have at least one basic (strict) subformula $\theta_1$. Then $\Phi$ is equivalent to the formula $\exists \kappa_1.(\Phi[\theta_1 \leftarrow \kappa_1] \land \text{AG}(\kappa_1 \leftrightarrow \theta_1))$, where $\kappa_1$ is a fresh atomic proposition, and $\varphi[\alpha \leftarrow \beta]$ is $\varphi$ where every occurrence of $\alpha$ is replaced by $\beta$. Indeed, any state reachable from the current state $x$ will be labelled by $\kappa_1$ if $\theta_1$ holds true at that state (NB: the states that are not reachable from $x$ do not matter for the truth value of $\Phi$), and this enforces the equivalence. We have $|S_{\Phi[\theta_1 \leftarrow \kappa_1]}| < |S_\Phi|$, thus we can apply induction hypothesis to get:

$$\Phi[\theta_1 \leftarrow \kappa_1] \equiv \exists \{\kappa_2 \ldots \kappa_m\}. (\Phi_0 \land \bigwedge_{i=2 \ldots m} \text{AG}(\kappa_i \leftrightarrow \theta_i)) = \Psi'$$

Where $\kappa_2 \ldots \kappa_m$ are fresh atomic propositions, $\Phi_0$ is a Boolean combination of basic CTL formulas and every $\theta_i$ is a basic CTL formula. We can conclude that:

$$\Phi \equiv \exists \kappa_1.[(\exists \{\kappa_2 \ldots \kappa_m\}. (\Phi_0 \land \bigwedge_{i=2 \ldots m} \text{AG}(\kappa_i \leftrightarrow \theta_i))) \land \text{AG}(\kappa_1 \leftrightarrow \theta_1)]$$

$$\equiv \exists \{\kappa_1 \ldots \kappa_m\}. (\Phi_0 \land \bigwedge_{i=1 \ldots m} \text{AG}(\kappa_i \leftrightarrow \theta_i)) = \Psi$$
Note that the last equivalence comes from the fact that no $\kappa_i$ with $i > 1$ occurs in $\theta_1$. By i.h. the size of $\Psi'$ is linear in $|\Phi[\theta_i \leftarrow \kappa_1]|$, and the size of $\Phi[\theta_i \leftarrow \kappa_1]$ is smaller than that of $\Phi$, therefore the size of $\Psi$ is linear in $|\Phi|$. ▲

It is important to note that $S_\Phi$ can be described as $\{\varphi_1 \ldots \varphi_m\}$, where $\varphi_1 = \theta_1$, and $\varphi_{i+1} = \theta_{i+1}[\kappa_1 \leftarrow \varphi_1, \ldots, \kappa_i \leftarrow \varphi_i]$. The correspondance $\theta_i \leftarrow \varphi_i$ will be crucial later on.

In the following, a QCTL formula of the form $\Q(\Phi_0 \land \bigwedge_{i=1}^m \AG(\kappa_i \leftarrow \theta_i))$, where $\Q$ is a sequence of quantifications, $\Phi_0$ is a Boolean combination of basic CTL formulas, every $\kappa_i$ is an atomic proposition, and every $\theta_i$ (for $i = 1, \ldots, m$) is a basic CTL formula, is said to be a flat formula.

As a corollary of previous results, we have:

**Proposition 10.** Any QCTL formula $\Phi$ is equivalent to some flat formula whose size is linear in $|\Phi|$.

Given a QCTL formula $\Phi$, we use $\mathit{flat}(\Phi)$ to denote the flat formula equivalent to $\Phi$, obtained by first translating $\Phi$ into Prenex normal form, and then transform the CTL subformula as described in Proposition 9.

Applying method FP to some flat formula provides a QBF formula of polynomial size since a flat formula has a temporal height less or equal to 2:

**Corollary 11.** Given a QCTL formula $\Phi$, a Kripke structure $\mathcal{K} = \langle V,E,\ell \rangle$ and a state $x$, the QBF formula $\mathit{fp}(\mathit{flat}(\Phi))$ obtained by applying the rules of Table 1 and the rule 8 is valid iff $\mathcal{K}, x \models \Phi$. And the size of $\mathit{fp}(\mathit{flat}(\Phi))$ is in $O(|V| \cdot (|V| + |E|) \cdot |\Phi|)$.

Therefore this reduction (called FFP) provides a PSPACE algorithm for QCTL model-checking. But there are two disadvantages to this approach. First, putting the formula into Prenex normal form may increase the number of quantified atomic propositions and the number of alternations (which is in fine linear in the number of quantifiers in the original formula) [10]. For example, when extracting a quantifier $\forall$ from some $\EX$ modality, we need to introduce two propositions, this can be seen for the formula $\EX(\forall p. (\AX p \lor \AX \neg p))$ which is translated as:

$$\exists z. \forall p. \forall z'. \left((\EX(z \land z') \Rightarrow \AX(z \Rightarrow z')) \land \EX(z \land (\AX p \lor \AX \neg p))\right)$$

where the proposition $z$ is used to mark a state, and $z'$ is used to enforce that only at most one successor is labelled by $z$. Of course, these two remarks may have a strong impact on the complexity of the decision procedure. Finally, note also that the resulting QBF formula is not in Prenex normal form.

### 3.4 Variant of FFP

In the previous reduction, the modalities $\EU$ and $\AU$ may introduce an alternation of quantifiers: an atomic proposition $\kappa$ is introduced by an existential quantifier, and then a universal quantifier introduces a variable $z$ to encode the fixed point characterisation of $U$. We propose another reduction in order to avoid this alternation: for this, we will use bit vectors (instead of single Boolean values) associated with every state to encode the distance from the current state to a state satisfying the right-hand side of the Until modality. This reduction produces a prenex QBF formula.
Proposition 12. Any QCTL formula \( \Phi \) is equivalent to some QCTL formula \( \Psi \) in NNF of the form: \( \Psi = Q \exists \{ \kappa_1 \ldots \kappa_m \}. (\Phi_0 \land \bigwedge_{i=1 \ldots m} \text{AG}(\kappa_i \Rightarrow \theta_i)) \) where \( Q \) is a sequence of quantifications, \( \Phi_0 \) is a CTL formula containing only the temporal modalities \( \text{EX}, \text{AX} \) or \( \text{AG} \) and whose temporal height is less or equal to 1, and every \( \theta_i \) is a basic CTL formula (with \( 1 \leq i \leq m \)). Moreover, \( |\Psi| \) is in \( O(|\Phi|) \).

From the previous proposition, we derive a new reduction to QBF (called FBV). For modalities \( \text{EW} \) and \( \text{AW} \), we use the same encoding as for method FP, except that we use the corresponding Boolean proposition \( \kappa_i \) directly for the greatest fixed point. And for Until-based modalities, corresponding to least fixed points, we will use bit vectors instead of atomic propositions to encode the truth value of \( \text{EU} \) or \( \text{AU} \): for a formula \( \theta_i = E \varphi U \psi \), we will consider a bit vector \( \vec{\kappa}_i \) of length \( \lceil \log(|V|+1) \rceil \) for every state instead of a single Boolean value \( \kappa_i \). The idea is that in a state \( x \), the value \( \vec{\kappa}_i \) encodes in binary the distance (in terms of number of transitions) from \( x \) to a state satisfying \( \psi \) along a path satisfying \( \varphi \). And for \( \theta_i = A \varphi U \psi \), the value \( \vec{\kappa}_i \) encodes the maximal distance before a state satisfying \( \psi \) (along a path where \( \varphi \) is true). In the following, such a \( \vec{\kappa}_i \) associated to a \( \theta_i \) based on an Until is called an Until-\( \kappa \). Note that given a bit vector \( \vec{\kappa}_i \) and an integer value \( d \) encoded in binary, we will use \( \vec{\kappa}_i^{\leq d} \) and \( \vec{\kappa}_i^{> d} \) to denote the corresponding propositional formulas over \( \vec{\kappa}_i \).

The new reduction is based on the rewriting rules of Table 3 for several operators, and we define the reduction for \( \text{AG}(\kappa_i \Rightarrow \theta_i) \) for \( \theta_i = E \varphi W \psi \) or \( \theta_i = A \varphi W \psi \) as follows:

\[
\begin{align*}
\hat{\varphi} \land \hat{\psi} &= \hat{\varphi} \land \hat{\psi} \\
\hat{\varphi} \lor \hat{\psi} &= \hat{\varphi} \lor \hat{\psi} \\
\hat{\varphi} \rightarrow \hat{\psi} &= \hat{\varphi} \rightarrow \hat{\psi} \\
\text{EX} \hat{\varphi} &= \bigvee_{(x,y) \in E} \hat{\varphi}^{x,y} \\
\text{AX} \hat{\varphi} &= \bigwedge_{(x,y) \in E} \hat{\varphi}^{x,y} \\
\text{AG} \hat{\varphi} &= \bigwedge_{(x,y) \in E^*} \hat{\varphi}^{x,y} \\
\hat{\varphi}_{P} &= \begin{cases} 
p \quad & \text{if } p \in P \text{ and } p \text{ is a Boolean var.} \\
p < |V| \quad & \text{if } p \in P \text{ and } p \text{ is a Bit vect.} \\
\top \quad & \text{if } p \notin P \land p \notin \ell(x) \\
\bot \quad & \text{otherwise}
\end{cases}
\end{align*}
\]

Table 3 Transformation rules for method FBV

First, we consider a QCTL formula \( \Phi \) under negation normal form (NNF), where the negation is only applied to atomic propositions. This transformation makes that \( \Phi \) is built from temporal modalities in \( S_{\text{mod}} = \{ \text{EX}, \text{AX}, \text{EU}, \text{AU}, \text{EW}, \text{AW} \} \)

We can then reformulate Proposition 10 as the following proposition, whose proof is in appendix B:

From the previous proposition, we derive a new reduction to QBF (called FBV). For modalities \( \text{EW} \) and \( \text{AW} \), we use the same encoding as for method FP, except that we use the corresponding Boolean proposition \( \kappa_i \) directly for the greatest fixed point. And for Until-based modalities, corresponding to least fixed points, we will use bit vectors instead of atomic propositions to encode the truth value of \( \text{EU} \) or \( \text{AU} \): for a formula \( \theta_i = E \varphi U \psi \), we will consider a bit vector \( \vec{\kappa}_i \) of length \( \lceil \log(|V|+1) \rceil \) for every state instead of a single Boolean value \( \kappa_i \). The idea is that in a state \( x \), the value \( \vec{\kappa}_i \) encodes in binary the distance (in terms of number of transitions) from \( x \) to a state satisfying \( \psi \) along a path satisfying \( \varphi \). And for \( \theta_i = A \varphi U \psi \), the value \( \vec{\kappa}_i \) encodes the maximal distance before a state satisfying \( \psi \) (along a path where \( \varphi \) is true). In the following, such a \( \vec{\kappa}_i \) associated to a \( \theta_i \) based on an Until is called an Until-\( \kappa \). Note that given a bit vector \( \vec{\kappa}_i \) and an integer value \( d \) encoded in binary, we will use \( \vec{\kappa}_i^{\leq d} \) and \( \vec{\kappa}_i^{> d} \) to denote the corresponding propositional formulas over \( \vec{\kappa}_i \).

The new reduction is based on the rewriting rules of Table 3 for several operators, and we define the reduction for \( \text{AG}(\kappa_i \Rightarrow \theta_i) \) for \( \theta_i = E \varphi W \psi \) or \( \theta_i = A \varphi W \psi \) as follows:
The quantifiers $\exists^1$ and $\forall^1$ are very useful in many specifications. It can be interesting to develop ad-hoc algorithms in order to improve the generated QBF formulas. For the first methods we described, they are translated into propositional formulas (instead of introducing extra quantified atomic propositions as they are formally defined). Another method consists in using bit vectors as in the treatment of Until modalities described above: in this case, these quantifiers introduce a unique bit vector of size $\lceil \log(|V| + 1) \rceil$ to store the number of the state.
selected by the quantifier (that is the state which will be labelled by the proposition). This method is interesting, since it reduces the number of quantified propositions, it is integrated in the method FBV.

4 Experimental results

In this section, we consider three examples to illustrate the QBF-based model-checking approach for QCTL. The problems we will consider can be solved more efficiently without a QCTL model-checker, but they provide valuable insights on the performance of the reduction strategies, and the properties to be checked cannot be expressed with classical temporal logics. The first example shows that the performances can change significantly when we choose different formulas to check, even if they are equivalent. The second problem illustrates how QCTL can naturally express properties related to game theory, and it is a situation where we can observe how bounded model-checking can improve the performance of FBV. In the last example, we check formulas with a large temporal height, thus we can see how much space is gained by flattening the formulas. We wanted to manipulate graphs that could easily be scaled up by tweaking a few parameters, so we chose to use grids.

We have implemented a prototype to try the different reduction strategies. Our tool is available online: given a Kripke structure $K$ with a state $x$ and a formula $\Phi$, it produces a specification file (corresponding to $\hat{\Phi}^x$) for the SMT-solver Z3 [5]. The choice of Z3 was motivated by the fact that the generated QBF formulas are not always Prenex, which many QBF-solvers require, unlike Z3.

4.1 $k$-connectivity

Here, we consider an undirected graph, and we want to check whether there exist (at least) $k$ internally disjoint paths from a vertex $x$ to some vertex $y$. A classical result in graph theory due to Menger ensures that, given two vertices $x$ and $y$ in a graph $G$, the minimum number of vertices whose deletion makes that there is no more paths between $x$ and $y$ is equal to the maximum number of internally disjoint paths between these two vertices.

We can encode these two ideas by the following QCTL formulas (interpreted over $x$):

$$\Phi_k = \exists p_1 \ldots \exists p_{k-1} \left( \bigwedge_{1 \leq i < k} \mathbf{EX}(E(p_i \land \bigwedge_{j \neq i} \neg p_j) \mathbf{U} y) \land \bigwedge_{1 \leq i < k} \mathbf{EX}(E(\neg p_i) \mathbf{U} y) \right)$$

(13)

$$\Psi_k = \forall^1 p_1 \ldots \forall^1 p_{k-1} \mathbf{EX}(E(\bigwedge_{1 \leq i < k} \neg p_i) \mathbf{U} y)$$

(14)

$\Phi_k$ uses the labelling by the $p_i$’s to mark the internal vertices of $k$ paths between the current position and the vertex $y$. The modality $\mathbf{EX}$ is used to consider only the intermediate states (and not the starting state). The formula $\Psi_k$ proceeds differently: the idea is to mark exactly $k-1$ states with $p_1, \ldots, p_{k-1}$ and to verify that there still exists at least one path leading to $y$ without going through the states labelled by some $p_i$.

By Menger’s Theorem, we know that these formulas are equivalent over undirected graphs.

---

6 NB: For reductions FFP and FBV the formula has to be given in Prenex normal form.
7 https://www.irif.fr/~francoisl/qctlmc.html
8 Two paths src $\leftrightarrow r_1 \leftrightarrow \ldots \leftrightarrow r_k \leftrightarrow$ dest and src $\leftrightarrow r'_1 \leftrightarrow \ldots \leftrightarrow r'_{k'} \leftrightarrow$ dest are internally disjoint iff $r_i \neq r'_j$ for any $1 \leq i \leq k$ and $1 \leq j \leq k'$. And note that with this def., if there is an edge $(x, y)$, there exist $k$ internally disjoint paths from $x$ to $y$ for any $k$. 
We interpret these formulas over Kripke structures $S_{n,m}$ with $n \geq m$ (see Figure 1) which correspond to two kinds of grids $n \times n$ connected by $m$ edges (these edges are of the form $(q_{i,n}, r_{1,i})$ or $(q_{n,i}, r_{i,1})$). The initial state is $q_{1,1}$ and when evaluating $\Phi_k$ or $\Psi_k$ we assume the state $r_{n,n}$ to be labelled by $y$. In this context, we clearly have that $\Phi_k$ and $\Psi_k$ hold for true at $q_{1,1}$ iff $k \leq m$.

![Figure 1](image-url) Structure $S_{n,m}$ for the $k$-connectivity problem.

Detailed results are presented in Appendix C. The main lessons we can see are:

- Formula $\Phi_k$ is much more difficult to verify: the number of temporal modalities is probably one explanation. Another one for methods FP and FFP (based on the fixed point characterization of $U/W$ modalities) could be an alternation of quantifiers: in $\Phi_k$, there is an existential quantification over the $p_i$s and the $EUs$ introduce a universal quantification, but it is not the case for $\Psi_k$, where there are only universal quantifications for these reductions.

- The reduction FP is the most efficient: it can be used to verify models with more than two thousands states when $m$ is small. The method FFP is also rather efficient, but the flattening seems to be too costly for such a simple formula.

- The reduction UU produces very large QBF formulas, but rather simple to check.

We can generalise the problem by verifying that there exist at least $k$ internally disjoint paths between any pair of reachable vertices $x$ and $y$ in a given structure. These previous formulas can be modified as follows:

$$\Phi_k^g = \forall y \exists p_1 \ldots \exists p_{k-1} \mathsf{AG} \left( \bigwedge_{1 \leq i < k} \mathsf{EX} \left( \mathsf{E} \left( \bigwedge_{j \neq i} \neg p_j \right) \cup y \right) \land \mathsf{EX} \left( \bigwedge_{1 \leq i < k} \neg p_i \right) \cup y \right)$$

$$\Psi_k^g = \forall y \forall p_1 \ldots \forall p_{k-1} \mathsf{AG} \left[ \mathsf{EX} \left( \mathsf{E} \left( \bigwedge_{1 \leq i < k} \neg p_i \right) \cup y \right) \right]$$

In that case, $\Phi_k^g$ is useless: too complex to be verified. And the method FP is still the most efficient.

### 4.2 Nim game

Nim game is a turn-based two-player game. A configuration is a set of heaps of objects and a boolean value indicating whose turn it is. At each turn, a player has to choose one
non-empty heap and remove at least one object from it. The aim of each player is to remove the last object. Given a configuration $c$ and a Player-$J$ with $J \in \{1, 2\}$, we can build a finite Kripke structure $S_J$, where $x_c$ is a state corresponding to the configuration $c$; and use a QCTL formula $\Phi^J_{\text{win}}$ such that $S, x_c \models \Phi^J_{\text{win}}$ iff Player-$J$ has a winning strategy from $c$. Note that there is a simple and well-known criterion over the numbers of objects in each heap to decide who has a winning strategy, but we consider this problem just because it is interesting to illustrate what kind of problem we can solve with QCTL.

Each configuration corresponds to a state in $S_J$. Every move for Player-$J$ from a configuration $c$ to a configuration $c'$ provides a transition $(x_c, x_{c'})$ in $S_J$. However, a move of Player-$J$ from $c$ to $c'$ is encoded as two transitions $x_c \rightarrow x_{c,c'} \rightarrow x_{c'}$ where $x_{c,c'}$ is then an intermediary state we use to encode a strategy for Player-$J$ (marking $x_{c,c'}$ by an atomic proposition will correspond to Player-$J$ choosing $c'$ from $c$). We assume that every state $x_c$ is labelled by $t_1$ if it’s Player-$1$’s turn to play at $c$, and by $t_2$ otherwise. Every intermediary state $x_{c,c'}$ is labelled by int. We also label empty configurations by $w_1$ or $w_2$, depending on which player played the last move.

Clearly, the size of $S$ will depend on the number of objects in each set in the initial configuration. The formula $\Phi^J_{\text{win}}$ depends only on $J$:

$$\Phi^J_{\text{win}} = \exists m. \left( \text{AG}(t_J \Rightarrow \text{EX}m) \land \text{AF}(w_J \lor (\text{int} \land \neg m)) \right)$$

This formula holds true in a state corresponding to some configuration $c$ iff there exists a labelling by $m$ such that every reachable configuration where it’s Player-$J$’s turn, has a successor labelled by $m$ (thus a possible choice to do) and every execution from the current state leads to either a winning state for Player-$J$ or a non-selected intermediary state, therefore all outcomes induced by the underlying strategy have to verify $Fw_J$. Note that in this example, the Kripke structure is acyclic (except the self-loops on the ending states).

From detailed results in appendix, we can see:

- One can consider structures with more than 10 thousand states. Note that the number of heaps is important for the size of the model, but the maximal length of a game depends on the number of objects (and is rather small in our examples).
- The most efficient method is FFP (with FP).
- Method FBV is more efficient when we consider bounded model-checking. Note that in this case, the verification may not be complete: if the QBF formula is valid, the property is satisfied by the structure, otherwise, no conclusion can be done, except if we can prove that the chosen bound was big enough to be sure that there is no solution. In our case, we can easily compute the maximal bound: at each turn, a player has to pick at least one object, such a move may give rise to one transition in the model (for the opponent), or two transitions (for the player for whom we look for a strategy). Thus, if there are $n$ objects in the initial configuration, we can choose $\frac{3n}{2}$ for the bound.

### 4.3 Resources distribution

The last example is as follows: given a Kripke structure $S$ and two integers $k$ and $d$, we aim at choosing at most $k$ states (called targets in the following) such that every reachable state (from the initial one) can reach a target in less than $d$ transitions. This problem can be encoded with the following QCTL formula where $d$ modalities $\text{EX}$ are nested:

$$\Phi_{\text{res}} = \exists c_1 \ldots \exists c_k \text{AG}\left( \bigvee_{1 \leq i \leq k} c_i \lor \text{EX}\left( \bigvee_{1 \leq i \leq k} c_i \lor \left( \ldots \lor \text{EX}\left( \bigvee_{1 \leq i \leq k} c_i \right) \right) \right) \right)$$
For experimental results, we consider the grid $K_{n,m}$ described at Figure 2. Note that for this example, reductions UU and FP are similar because there is no Until in the formula ($AG$ is treated as a conjunction). From detailed results in appendix, one can see:

- Only small models have been successfully verified.
- The nesting of $EX$s operators give an advantage to the reduction based on flattening (FBV and FFP): the size of the QBF formula increases more slowly.
- The reduction FBV is the most efficient on this example. Since there is no Until modality (except behind $AG$ which is treated separately), the difference with FFP is due to the encoding of $\exists 1$ operator with a unique bit vector in FBV, this choice seems to be more efficient in this example.

5 Conclusion

We have presented several reductions from QCTL model-checking to QBF. This provides a first tool for QCTL model-checking. Of course, this is an ongoing-work, and many improvements are possible: the reduction strategies are still naive and could be significantly improved, and a better understanding of QBF-solvers would also be helpful to produce more efficient formulas (we have not yet tried to normalise formulas in a specific form which is often a crucial aspect in SAT/QBF-solving). Still, these first results are rather interesting and encouraging. They show the importance of writing "good" QCTL formulas for which the solver will be able to provide a result (this problem already exists for classical temporal logics, but it is more significant here due to the complexity induced by the quantifications). The examples also show that there is no 'one best strategy': it depends on the structure of the considered formula, and then offering several reduction strategies seems to be necessary in a QBF-based model-checker for QCTL. Finally this work is also interesting because it could easily be adapted for other logics (like Sabotage logics [17]). In particular, we plan to figure how it could be adapted for LTL. In the future, we plan to continue to work on reduction strategies, and to use other QBF-solvers.
References


A. Hossain and F. Laroussinie 23:17


A Proof of Theorem 4

Proof. We assume Φ to be fixed and we prove the property by structural induction over ϕ. Boolean operators are omitted.

ϕ = p: if K, x |= e p, then either p is a quantified proposition and it belongs to dom(ε) and x belongs to ε(p) and thus vε |= p by def. of vε, or p belongs to ℓ(x). In both cases we have vε |= ˆp|dom(ε). The converse is similar.

ϕ = EXψ: K, x |= e EXψ iff there exists (x, x′) ∈ E s.t. K, x′ |= e ψ, iff (by i.h.) there exists (x, x′) ∈ E s.t. vε |= ψ′,|dom(ε) which is equivalent to vε |= EXψ|dom(ε). The definition of ψ′ corresponds to a finite unfolding of the expansion law that characterizes the EU modality. Assume K, x |= e ϕ. There exists a path ρ ∈ PathK(x) and a position i ≥ 0 s.t. ρ(i) |= ε ψ2 and ρ(k) |= ε ψ1 for any 0 ≤ k < i. The finite prefix x = ρ(0) . . . ρ(k) can be assumed to be simple, and then k < |V|. By using i.h., we get vε |= ψ2|dom(ε), and vε |= ψ1|dom(ε) for any 0 ≤ k < i. From this point, the reader can easily verify by induction (starting at i, down to 0) that vε |= ψ1Uψ2|dom(ε) for all k ≤ i. This makes ϕ|dom(ε) to be satisfied by vε. Conversely, assume vε |= ϕ|dom(ε). Let us build a finite path x = x0, x1, . . . xN that satisfies ψ1Uψ2. The first vertex is of course x, and we have vε |= ψ1Uψ2|dom(ε). Fix i so that for every 0 ≤ k ≤ i, xk is built, vε |= ψ1Uψ2|dom(ε), and (xk, xk+1) ∈ E when k < i. Then, there must be (xg, y) ∈ E so that y /∈ {xj | j ≤ i}, and vε |= ψ1Uψ2|dom(ε) for every 0 ≤ k < i. By using i.h., we get K, x |= e ϕ.

ϕ = Aψ1Uψ2: this case is similar to the previous one, except that we have to consider loops. Assume K, x |= e ϕ. Then any path issued from x satisfies ψ1Uψ2. If x contains a self-loop, then ψ2 has to be satisfied at x (this is ensured by the first case in the def. of ϕ|dom(ε)). Otherwise we consider all the paths from x: either there is a simple prefix witnessing ψ1Uψ2, or there is a loop from some point. In the latter case, one of the state in the loop has to verify ψ2. In both cases, the definition of ϕ|dom(ε) gives the result.
B Proof of Proposition 12

Proof. Consider w.l.o.g. a QCTL formula $\Phi$ in Prenex normal form and NNF. We can define $\Phi_0$ and the basic formulas $\theta_i$s approximately as in Proposition 9, except that $\Phi_0$ contains only $\text{EX}$, $\text{AX}$ or $\text{AG}$ modalities, and every other modality gives rise to some quantified proposition $\kappa$ and a subformula $\text{AG}(\ldots)$ in the main conjunction of $\Psi$. Every $\theta_i$ starts with a modality in $S_{\text{mod}}$. Let $\varphi_i$ be the original $\Phi$-subformula associated with $\theta_i$. Note that $\Psi$ is in NNF, and $\kappa_i$ occurs only once in $\Psi$ in the scope of a negation, and it happens in the subformula $\text{AG}(\kappa_i \Rightarrow \theta_i)$. We now have to show that $\Phi$ is equivalent to $\Psi$. Consider the formula $\tilde{\Psi}$ where every $\Rightarrow$ is replaced by $\Leftrightarrow$: by following the same arguments of Proposition 9, we clearly have $\Phi \equiv \tilde{\Psi}$, and $\Phi \Rightarrow \tilde{\Psi}$. It remains to prove the opposite direction.

To prove $\Psi \Rightarrow \tilde{\Psi}$, it is sufficient to show that this is true for the empty $Q$ (as equivalence is substitutive). Assume $K, x \models_{\epsilon} \Psi$. Then there exists an environment $\epsilon'$ from $\{\kappa_1, \ldots, \kappa_m\}$ to $2^V$ such that $K, x \models_{\epsilon'} \Phi_0 \land \bigwedge_i \text{AG}(\kappa_i \Rightarrow \theta_i)^9$. Now we have:

$$\forall i, \ K, x \models_{\epsilon'} \theta_i \Rightarrow K, x \models_{\epsilon} \varphi_i \quad \text{and} \quad K, x \models_{\epsilon'} \Phi_0 \Rightarrow K, x \models_{\epsilon'} \tilde{\Phi}_0$$

Indeed, assume that it is not true and $K, x \not\models_{\epsilon'} \theta_i$ and $K, x \not\models_{\epsilon} \varphi_i$. Consider such a formula $\varphi_i$ with the smallest temporal height. The only atomic propositions $\kappa_j$ occurring in $\theta_i$ are then associated with some $\theta_j$ and $\varphi_j$ which verify the property and thus any state satisfying such a $\theta_j$, also satisfies $\varphi_j$. Therefore any state labelled by such a $\kappa_j$ is correctly labelled (and satisfies $\varphi_j$). And the states that are not labelled by $\kappa_j$ cannot make $\theta_i$ to be wrongly evaluated to true (because $\kappa_j$ is not in the scope of a negation). Therefore $\varphi_i$ holds true at $x$. The same holds for $\Phi_0$ and $\tilde{\Phi}_0$. As a direct consequence, we have $\Psi \Rightarrow \tilde{\Psi}$.

C Experimental results

Detailed results for the three examples are presented in this section. In every case, we distinguished the time required to build the QBF formula (the Z3 specification) and the time required to solve it. Times are given in seconds.

---

9 We define the graph $G_{\epsilon \circ \epsilon'}$ of $\epsilon \circ \epsilon'$ to be $G_{\epsilon} \equiv G_{\epsilon'}$. 
### C.1 $k$-connectivity

<table>
<thead>
<tr>
<th>n, m</th>
<th>formula</th>
<th>$\Psi_2$</th>
<th>$\Psi_4$</th>
<th>$\Phi_4$</th>
<th>$\Psi_5$</th>
<th>$\Psi_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td># states</td>
<td>res</td>
<td>sat</td>
<td>unsat</td>
<td>sat</td>
<td>unsat</td>
<td>unsat</td>
</tr>
<tr>
<td>Time to build z3 form.</td>
<td>0.01</td>
<td>38.45</td>
<td>70.15</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Size of z3 form.</td>
<td>7318</td>
<td>7922175</td>
<td>29983034</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Time to solve z3 form.</td>
<td>0.04</td>
<td>2.38</td>
<td>8.61</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>

**Reduction UU**

| Time to build z3 form. | 0 | 0.03 | 0.07 | 0.09 | 388 |
| Size of z3 form. | 1313 | 9097 | 11234 | 27416 | 38675099 |
| Time to solve z3 form. | 0.03 | 0.06 | – | 0.16 | 132.67 |

**Reduction FP**

| Time to build z3 form. | 0.03 | 0.1 | 0.25 | 0.1 | – |
| Size of z3 form. | 3666 | 20944 | 56196 | 61519 | – |
| Time to solve z3 form. | 0.12 | 0.28 | – | 57.96 | – |

**Reduction FFP**

| Time to build z3 form. | 0.01 | 1 | 0.23 | 1.3 | – |
| Size of z3 form. | 6363 | 31243 | 107120 | 87681 | – |
| Time to solve z3 form. | 40.16 | – | 43.09 | – | – |

<table>
<thead>
<tr>
<th>n, m</th>
<th>formula</th>
<th>$\Psi_2$</th>
<th>$\Psi_4$</th>
<th>$\Phi_4$</th>
<th>$\Psi_5$</th>
<th>$\Psi_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td># states</td>
<td>res</td>
<td>unsat</td>
<td>sat</td>
<td>unsat</td>
<td>unsat</td>
<td>unsat</td>
</tr>
<tr>
<td>Time to build z3 form.</td>
<td>1.08</td>
<td>9.3</td>
<td>–</td>
<td>–</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size of z3 form.</td>
<td>1195684</td>
<td>8632341</td>
<td>–</td>
<td>–</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time to solve z3 form.</td>
<td>0.43</td>
<td>8.96</td>
<td>–</td>
<td>–</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Reduction UU**

| Time to build z3 form. | 0.02 | 0.04 | 1.94 | 190.94 |
| Size of z3 form. | 13268 | 18337 | 1698025 | 139992889 |
| Time to solve z3 form. | 0.1 | 0.23 | 45.51 | – |

**Reduction FP**

| Time to build z3 form. | 0.02 | 0.04 | 2.07 | 192 |
| Size of z3 form. | 5180 | 6877 | 551447 | 44643959 |
| Time to solve z3 form. | 0.10 | 35.38 | 180 | – |

**Reduction FFP**

| Time to build z3 form. | 0.1 | 0.05 | 0.95 | 195 |
| Size of z3 form. | 6285 | 8331 | 726681 | 59447253 |
| Time to solve z3 form. | 0.35 | – | – | – |
C.2 Nim game

<table>
<thead>
<tr>
<th># states</th>
<th>[2, 2]</th>
<th>[3, 2]</th>
<th>[4, 5, 2]</th>
<th>[3, 4, 5]</th>
<th>[2, 3, 4, 4]</th>
<th>[5, 4, 3, 6]</th>
<th>[2, 4, 8, 14]</th>
</tr>
</thead>
<tbody>
<tr>
<td>res</td>
<td>unsat</td>
<td>sat</td>
<td>unsat</td>
<td>sat</td>
<td>unsat</td>
<td>unsat</td>
<td>unsat</td>
</tr>
<tr>
<td>Time to build z3 form.</td>
<td>0.00</td>
<td>0.00</td>
<td>0.09</td>
<td>0.31</td>
<td>8.99</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Size of z3 form.</td>
<td>32</td>
<td>105</td>
<td>90095</td>
<td>285505</td>
<td>1011324</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Time to solve z3 form.</td>
<td>0.02</td>
<td>0.03</td>
<td>0.09</td>
<td>0.10</td>
<td>0.39</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

C.3 Resources distribution

<table>
<thead>
<tr>
<th>n × m</th>
<th>10 × 5</th>
<th>10 × 5</th>
<th>10 × 5</th>
<th>10 × 7</th>
<th>10 × 10</th>
<th>10 × 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>k, d</td>
<td>2, 8</td>
<td>4, 4</td>
<td>4, 6</td>
<td>4, 5</td>
<td>2, 3</td>
<td>4, 7</td>
</tr>
<tr>
<td># states</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>70</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>res</th>
<th>unsat</th>
<th>sat</th>
<th>unsat</th>
<th>unsat</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to build z3 form.</td>
<td>0.03</td>
<td>0.08</td>
<td>0.07</td>
<td>0.10</td>
<td>0.13</td>
</tr>
<tr>
<td>Size of z3 form.</td>
<td>112155</td>
<td>32357</td>
<td>70757</td>
<td>74417</td>
<td>45905</td>
</tr>
<tr>
<td>Time to solve z3 form.</td>
<td>0.08</td>
<td>274.00</td>
<td>0.06</td>
<td>–</td>
<td>0.09</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reduction UU − FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to build z3 form.</td>
</tr>
<tr>
<td>Size of z3 form.</td>
</tr>
<tr>
<td>Time to solve z3 form.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reduction FFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to build z3 form.</td>
</tr>
<tr>
<td>Size of z3 form.</td>
</tr>
<tr>
<td>Time to solve z3 form.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reduction FBV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to build z3 form.</td>
</tr>
<tr>
<td>Size of z3 form.</td>
</tr>
<tr>
<td>Time to solve z3 form.</td>
</tr>
</tbody>
</table>